Unix 下分布式编译环境的构建

Report Date: *2009-07-20*

Reporter Name: Walter Zhou

FUJI Xerox

Overview

Abstract:

- 用开源工具distcc来搭建分布式编译环境的背景
- 需要的软硬件条件
- distcc工作原理
- distcc配置简介
- LCW项目中应用分布式编译的效果

Scope:

- Unix OS
- C / C++
- Compiling

Key Word:

- Unix OS, C++, Distributed Compiling，distcc

Background

在LCW项目中，单机硬件条件不理想，c/c++中头文件设置的不妥，
造成Solaris下整个模拟器的编译要差不多5,6小时

1. 低端或淘汰的Sun工作站有7，8台
2. c/c++中头文件是FX提供的工具生成，不太可能人工去调整

编译慢的原因如下：

- 工作站CPU频率较低（属于淘汰货），内存为256M或512M
- 由于被编译器预处理后的c/c++代码很多都达到每个文件近10万LOC
- 在一台工作站上编译是串行编译

基于如此硬件条件，考虑发扬“人多力量大”的中国传统(人海战术)来加速编译过程，实现“**机多编译快**”的美好想法。

Exercise to reinforce learning if possible

distcc简介

distcc是Unix下的开源的分布式编译工具

http://distcc.samba.org/

原理：

把编译任务分派到指定的各个机器上去分别编译，并收集编译结果，最后在主控机上实现链接（链接是不能分布式的）

大项目一般都有成千上万的源码文件，而在**C/C++**中每个**c**或**cpp**文件是独立的编译单元（即独立完成词法分析，语法分析，语法树构建，中间代码产生，代码优化等），所以完全可以在不同环境不同机器上进行编译

distcc简介续

distcc的任务

- 主控机上的distcc把各个编译单元分发到其他参与编译的开发机上
- 参与编译的开发机上的distcc负责具体的编译并把结果发回给主控机

distcc的安装

1. 下载源码包，比如http://distcc.samba.org/ftp/distcc/distcc-2.18.3.tar.bz2

2. 在Unix下常规开源工具的安装步骤如下
   - tar jxvf distcc-2.18.3.tar.bz2
   - cd distcc-2.18.c
   - ./configuration
   - make
   - make install

3. 在主控机与其他编译机上都请按上面步骤安装distcc

4. 生成的3个可执行文件

```
wzhou@C2D008:/usr/local/bin % ls -l distcc*
-rwxr-xr-x 1 root root 199076 2009-03-13 15:35 distcc
-rwxr-xr-x 1 root root 188260 2009-03-13 15:35 distccd
-rwxr-xr-x 1 root root  81224 2009-03-13 15:35 distccmon-text
```

distcc的配置

（distcc只是分派编译任务的工具，具体编译还是依赖于编译器，比如gcc,这里gcc等编译工具的配置，不做介绍了。假设都没有问题。）

■ 主控机上个人Profile的设置

比如wzhou用户使用cshell，则请在HOME目录的.cshrc中通过引入**DISTCC_HOSTS**环境变量来加入参与编译的其他机器，组成编译器农场（Compile Farm）

■ 主控机上在编译项目的Makefile中请修改CC变量

```
wzhou@C2D001:/export/home/wzhou/lighttpd-1.4.19 % diff Makefile Makefile.org
85c85
< CC = distcc gcc
---
> CC = gcc
88c88
< CPP = distcc gcc -E
---
> CPP = gcc -E
162c162
< ac_ct_CC = distcc gcc
---
> ac_ct_CC = gcc
```

distcc的配置(续)

■ 辅助编译机上的配置

启动distcc的后台进程

distccd --daemon --allow 13.187.243.0/24

--daemon以后台进程运行

--allow是列出允许接受哪台主控机发来的编译任务

```
wzhou@C2D008:/export/home/wzhou % distccd --daemon --allow 13.187.243.71
wzhou@C2D008:/export/home/wzhou % ps -ef | grep distccd
   wzhou  8415  8413   0 10:45:41 ?           0:00 distccd --daemon --allow
13.187.243.71
   wzhou  8418  8402   0 10:45:55 pts/5       0:00 grep distccd
   wzhou  8413     1   0 10:45:40 ?           0:00 distccd --daemon --allow
13.187.243.71
   wzhou  8414  8413   0 10:45:40 ?           0:00 distccd --daemon --allow
13.187.243.71
   wzhou  8416  8413   0 10:45:42 ?           0:00 distccd --daemon --allow
13.187.243.71
```

distcc编译效果

Distcc是Unix下开源的分布式编译工具，它可以把本机上的编译任务分派到指定的其他机器上去编译来加速整个编译过程。或许这种方法能够加速我们项目LCW的编译过程。

我挑了我们的5台机器来做实验（都需要专为分布式编译而配置）。

C2D001，C2D008,C2D009,C2D012,C2D013 （这几台机器的配置各异，有差不多10年前买的机器，有4，5年前买的，有1年前买的）

其中C2D001是主控机，它将负责分派编译任务并完成最后的链接工作，而其他机器则接受C2D001的编译任务，完成分配给它的任务。

（我这种安排不是很科学，因为主控机C2D001是配置最差的机器，而它的责任与负担又最重，但这毕竟只是试验而已，所以就将就了吧）

我挑了一个小型软件lighttpd(轻量级Web Server)来测试传统编译与分布式编译在编译时间上的差异。

distcc编译效果（续）

分布式编译命令如下：

date; make -j 16; date （在单机上其实你也可以用-j参数，但意义不大，因为毕竟你就一块CPU；我在这里指定16，大概每个节点有4个文件在编译）

分布式编译时间统计：（编译了3次）

1.

Fri Mar 13 17:32:02 CST 2009

Fri Mar 13 17:35:05 CST 2009

183秒

2.

Fri Mar 13 17:37:04 CST 2009

Fri Mar 13 17:40:02 CST 2009

178秒

3.

Fri Mar 13 17:41:00 CST 2009

Fri Mar 13 17:43:59 CST 2009

179秒

编译时间比较稳定，（183 + 178 + 179） / 3 = 180秒

distcc编译效果（续）

传统编译命令如下：

Date; make; date

传统编译时间统计：（编译了3次）

1.

Fri Mar 13 17:50:15 CST 2009

Fri Mar 13 17:57:07 CST 2009

412秒

2.

Fri Mar 13 17:58:19 CST 2009

Fri Mar 13 18:05:16 CST 2009

417秒

3.

Fri Mar 13 18:07:27 CST 2009

Fri Mar 13 18:14:17 CST 2009

410秒

编译时间比较稳定，（412 + 417 + 410） / 3 = 413秒

distcc编译效果（续）

413 / 180 = 229%

性能的提高是及其明显的，几乎勿容置疑。

distcc编译效果（续）

■ 在LCW项目上的应用

在构建一个由淘汰机器组成的5机编译器农场（Compile Farm）下，整个LCW模拟器的clean build大约为2小时，而在任何一台单机上编译都要5到6小时。

最快的一次曾经只花了50分钟。

11：39分开始

12：29分结束

可能是快到吃饭时间了，Compile Farm中的机器空闲下来，所以可以全力参与编译。

所以整体编译的快慢还依赖于参与编译的各机器的繁忙程度。

监控分布式编译

当分布式编译时可以用distccmon-text命令监控到整个编译过程：(后面的主机名就是其被分派到编译任务的机器)

```
wzhou@C2D001:/export/home/wzhou % distccmon-text 2
 6395  Compile    mod_trigger_b4_dl.c          C2D008[0]
 6479  Compile    mod_magnet_cache.c           C2D008[1]
 6476  Compile    mod_cml.c                    C2D008[3]
 6423  Compile    mod_staticfile.c             C2D009[0]
 6580  Compile    lemon.c                      C2D009[1]
 6460  Compile    mod_cgi.c                    C2D009[2]
 6435  Compile    mod_dirlisting.c             C2D012[1]
 6478  Compile    mod_webdav.c                 C2D012[3]
 6436  Compile    mod_scgi.c                   localhost[1]
 6580  Compile    lemon.c                      C2D009[1]
 6460  Compile    mod_cgi.c                    C2D009[2]
 6435  Compile    mod_dirlisting.c             C2D012[1]
 6478  Compile    mod_webdav.c                 C2D012[3]
 6436  Compile    mod_scgi.c                   localhost[1]
 6580  Compile    lemon.c                      C2D009[1]
 6460  Compile    mod_cgi.c                    C2D009[2]
 6435  Compile    mod_dirlisting.c             C2D012[1]
 6478  Compile    mod_webdav.c                 C2D012[3]
 6436  Compile    mod_scgi.c                   localhost[1]
```

。。。。。。

Troubleshooting 1

g++: **Internal error**: Segmentation Fault (program cc1plus)

Please submit a full bug report.

See <URL:http://gcc.gnu.org/bugs.html> for instructions.

distcc[2054] ERROR: compile ../../application/copy//src/CMuLcwAppCpImsDtl.cpp on C2D005 failed

只有在分布式的情况下才会报错，在单机编译时不会，则那很有可能不是真正的错，而是机器配置比较低而引起的错 。

出错的原因是被编译的源文件在被预处理后达到9万多行（非常接近10万行），同时该文件又被派发到一些老机器上，比如上面的C2D005，该机器的内存是256M，配置比较低。

由于C2D005等老机器在为多位LCW开发者（目前是4位）服务，即该机可能同时在编译多个超大的源码文件。如果你关注过g++在编译LCW的大文件时用到的虚拟内存与物理内存的话，会注意到一般单个文件就要耗费内存在230M/200M之间（虚拟内存230M，物理内存200M，这之间的差值自然到交换文件中去了）。如果有多个大文件同时参与编译，就会出现g++在编译时申请内存失败而报告g++内部出错的情况（也就是上面出现的错）。

目前这种情况很难解决，因为distcc(分布式编译前端管理器)没有也不支持相关有条件派发编译任务，即根据compile farm中的机器配置的不同而调整派发优先级。

Troubleshooting 2

有时，我们会看到下面的报错信息：

distcc[8789] (dcc_select_for_read) ERROR: IO timeout

distcc[8789] (dcc_r_token_int) ERROR: read failed while waiting for token "DONE"

distcc[8789] Warning: failed to distribute ../../application/copy//src/CMuLcwAppCpCb.cpp to C2D009, running locally instead

distcc[8426] (dcc_select_for_read) ERROR: IO timeout

distcc[8426] (dcc_r_token_int) ERROR: read failed while waiting for token "DONE"

distcc[8426] Warning: failed to distribute ../../framework/widget/src/CMuWDTFeatureAccessButtonSetImpl.cpp to C2D012, running locally instead

distcc[8814] (dcc_select_for_read) ERROR: IO timeout

distcc[8814] (dcc_r_token_int) ERROR: read failed while waiting for token "DONE"

distcc[8814] Warning: failed to distribute ../../application/copy//src/CMuLcwAppCpOriDirBook.cpp to C2D012, running locally instead

C2D009和C2D012在某段时间突然很忙，以至于主控机C2D008等待对方编译文件超时，所以在只能在本机（C2D008）编译该文件了。

如果辅助编译机本身就很忙的话，分布式编译的效果会大打折扣

Walter Zhou

Walter.Zhou@chn.fujixerox.com